UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

5DATA005C.1 Data Engineering

COURSEWORK

Lecturer Name:

Ms Yaalini Balathasan

Name : ***Shahly Fayeek***

UOW ID : ***20521398***

IIT ID : ***20231076***

Table of Contents

Data Extraction

Sample CSV files that display the performance and sales statistics of a major retail store were downloaded from https://www.kaggle.com/. The dataset was subsequently utilized for additional analysis and transformation.

The following URL will take you to the superstore data set:
https://www.kaggle.com/datasets/bhanupratapbiswas/superstore-sales

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns

[ ] #mounting drive
    from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

Data Extraction

[ ] Customer = pd.read_csv("/content/drive/MyDrive/datasets/Customer.csv")
    Product = pd.read_csv("/content/drive/MyDrive/datasets/Product.csv")
    Order = pd.read_csv("/content/drive/MyDrive/datasets/Order.csv")
```

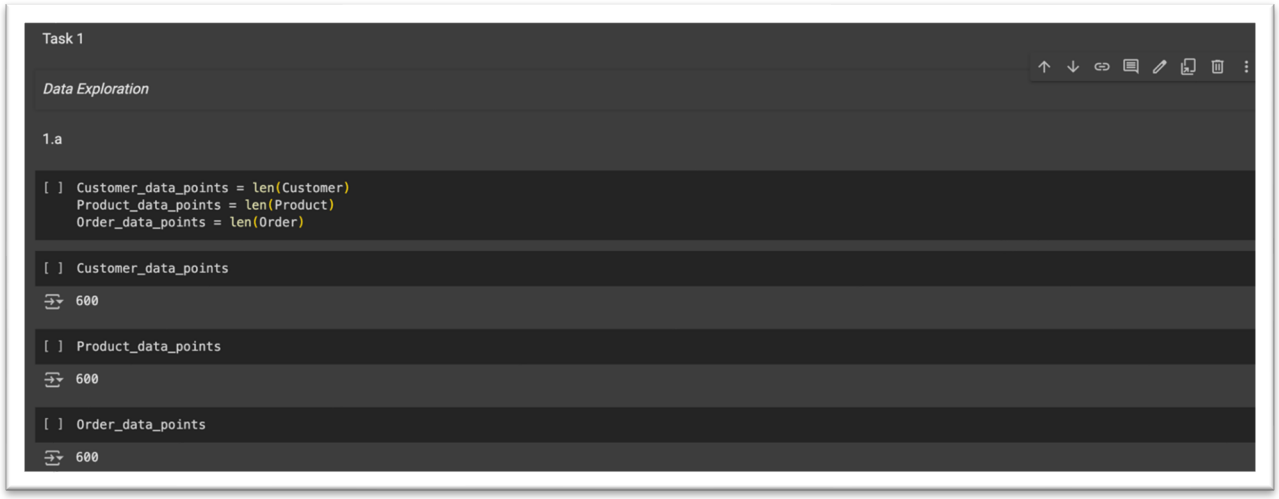
The data set was stored in my personal google drive in the “datasets” folder and it included information about the customer, the item he bought, and the particulars of his order. Three categories—Customer, Product, and Order—were created from the dataset. Customer_ID, Product_ID, and Order_ID are just a few examples of the unique attributes that were used to record the data. At least five attributes are present in each dataset; the Order dataset now has the additional attribute "Status". The pd.read_csv function, which supplied the file location, was used to extract (import) the datasets.

Task 1: Data Exploration

The dataset was reviewed and analyzed to gain an understanding of its structure and content.

(a) Number of data points

The Customer, Product, and Order CSV files each contain 600 records, as determined using the ‘len’ function in pandas.

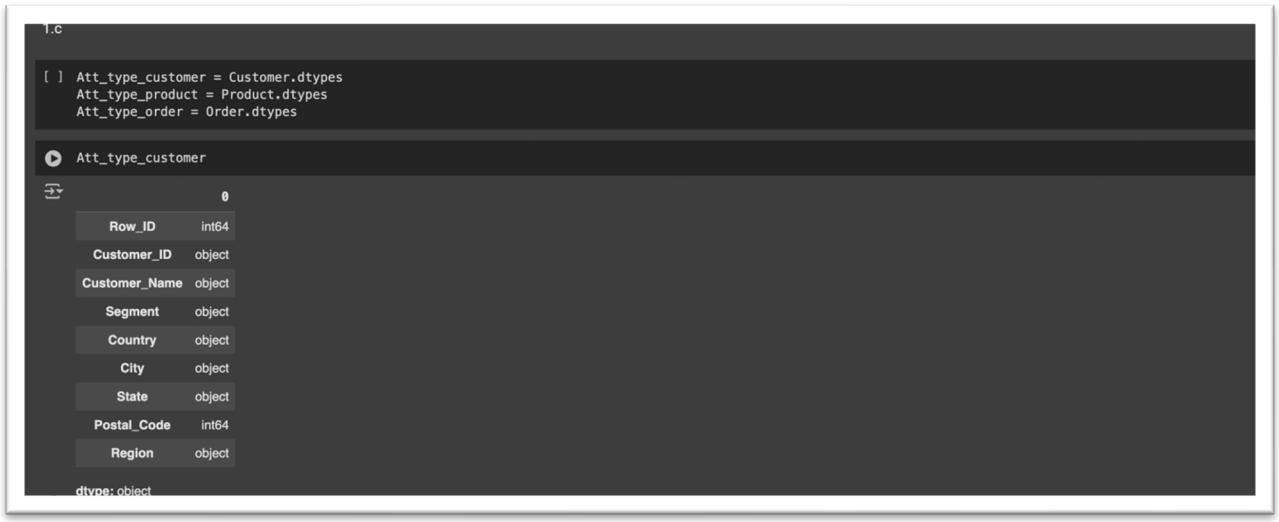


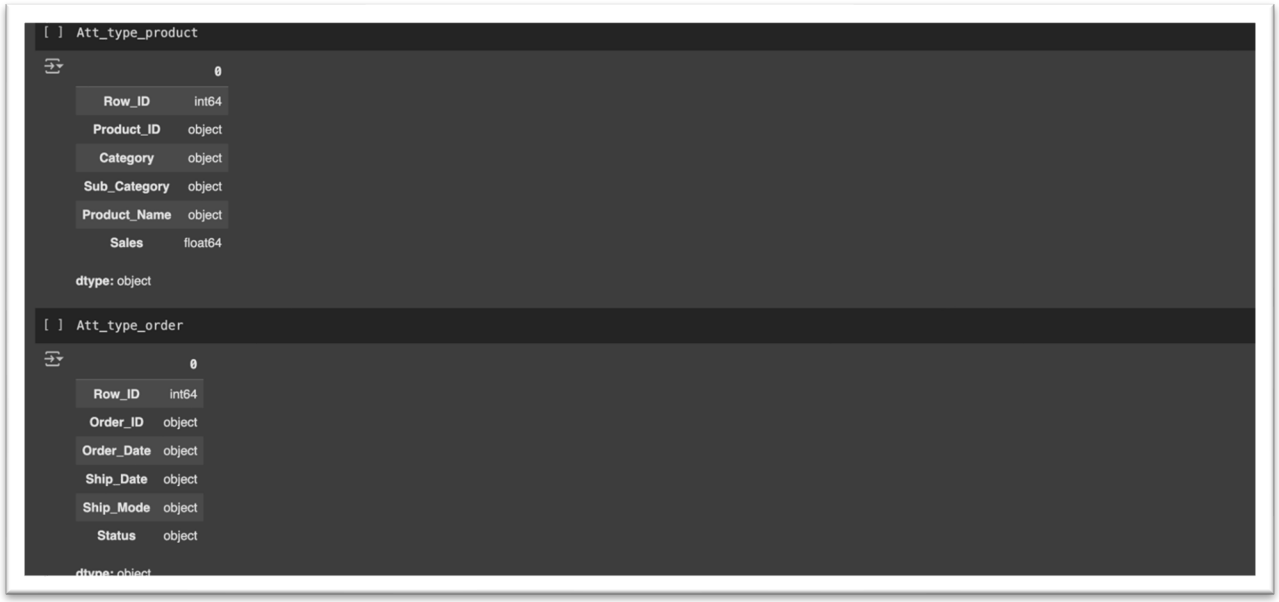
(b) Name of attributes

The attributes of each dataset were retrieved using the ‘list’ function.

(c) Type of attributes

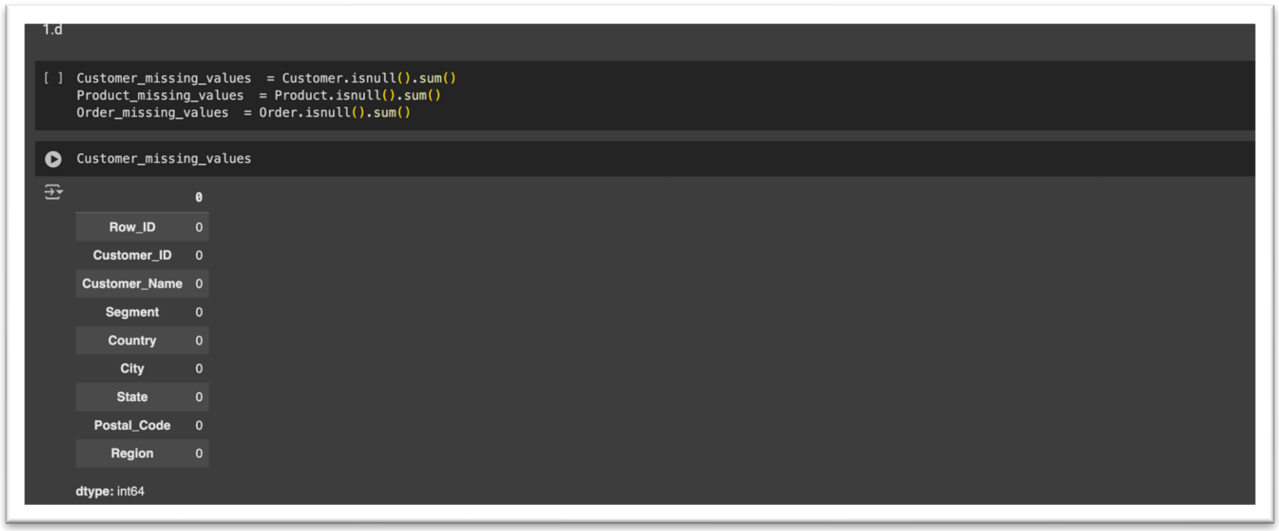
The data type of each attribute in the datasets was identified using the '.dtypes' function. It is evident that many attributes are of type 'object,' while only a few are integers or floats. Among these, only 'Row_ID', 'Postal_Code', and 'Sales' contain exclusively integer or float data types.

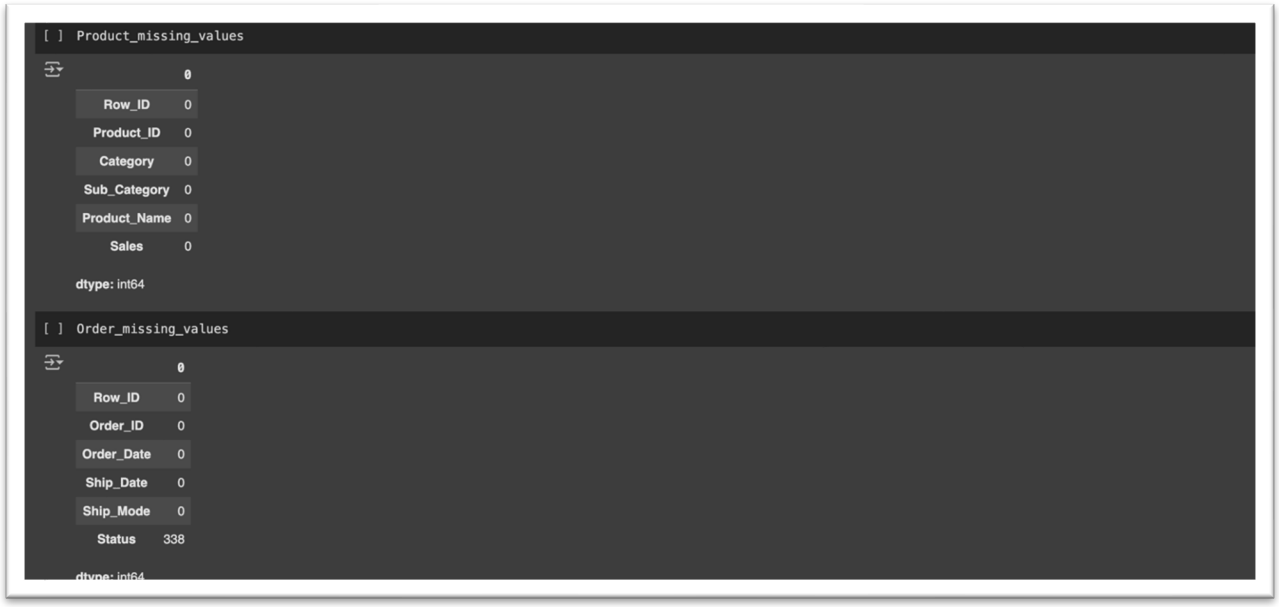




(d) Number of missing vales for each attribute

In the Customer and Product datasets, no missing values were found. However, the Order dataset contains 338 missing values, all of which are in the *Status* attribute. The exact count of missing values in each dataset is shown in the images above, and the total was calculated using the '.isnull()' function.





(e) Entry errors for each attribute

The ".unique()" function in Pandas is used to find all distinct (non-duplicate) values in specified column or series. It returns an array containing each unique value exactly once, in the order they first appear in the data.

1.e

```
[ ] Missing_values_error_customer  = Customer.isnull().sum()
    MissingValues_error_product    = Product.isnull().sum()
    MissingValues_error_order   = Order.isnull().sum()
```

```
[ ] Missing_values_error_customer
```

	0
Row_ID	0
Customer_ID	0
Customer_Name	0
Segment	0
Country	0
City	0
State	0
Postal_Code	0
Region	0

dtype: int64

```
[ ] print(Customer['State'].unique())
```

```
['Kentucky' 'California' 'Florida' 'North Carolina' 'Washington' 'Texas'
 'Wisconsin' 'Utah' 'Nebraska' 'Pennsylvania' 'Illinois' 'Minnesota'
 'Michigan' 'Delaware' 'Indiana' 'New York' 'Arizona' 'Virginia'
 'Tennessee' 'Alabama' 'South Carolina' 'Oregon' 'Colorado' 'Iowa' 'Ohio'
 'Missouri' 'Oklahoma' 'New Mexico' 'Louisiana' 'Connecticut' 'New Jersey'
 'Massachusetts' 'Georgia' 'Nevada' 'Rhode Island' 'Mississippi'
 'Arkansas' 'Montana']
```

```
[ ] print(Customer['Region'].unique())
```

```
['South' 'West' 'Central' 'East']
```

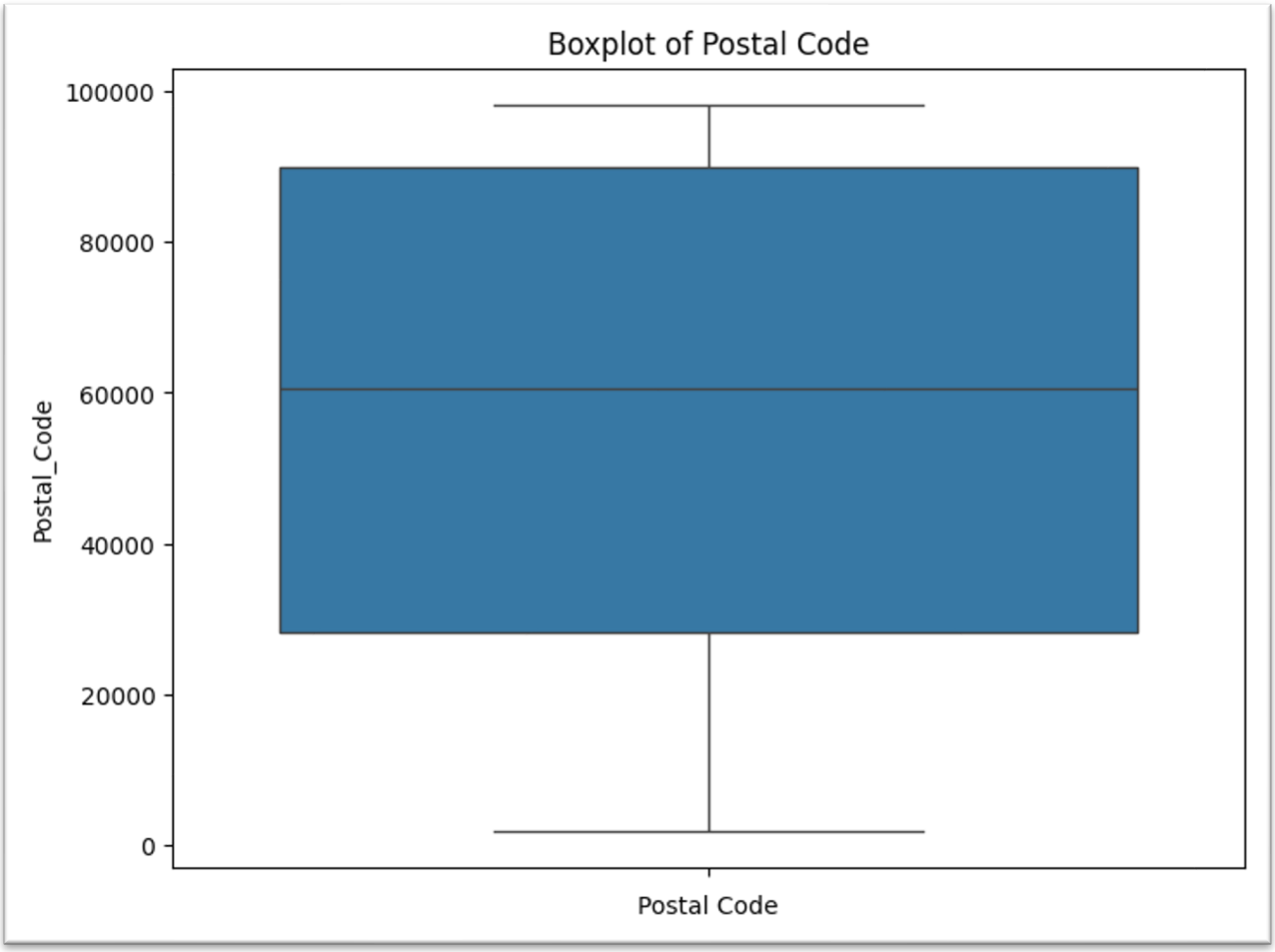
```
[ ] print(Customer['Segment'].unique())
```

```
['Consumer' 'Corporate' 'Home Office']
```

```
[ ] plt.figure(figsize=(8, 6))
    sns.boxplot(data=Customer, y='Postal_Code')
    plt.title('Boxplot of Postal Code')
    plt.xlabel('Postal Code')
    plt.show()
```

The below boxplot shows that there are no outliers in the ‘Postal_Code’ column of the ‘Customer’ dataset. Matplotlib and Seaborn are the libraries that were used to plot the diagram.

```
[ ] plt.figure(figsize=(8, 6))
    sns.boxplot(data=Customer, y='Postal_Code')
    plt.title('Boxplot of Postal Code')
    plt.xlabel('Postal Code')
    plt.show()
```



```
[ ] MissingValues_error_product
```

	0
Row_ID	0
Product_ID	0
Category	0
Sub_Category	0
Product_Name	0
Sales	0

dtype: int64

```
[ ] print(Product['Category'].unique())
['Furniture' 'Office Supplies' 'Technology']

[ ] print(Product['Sub_Category'].unique())
['Bookcases' 'Chairs' 'Labels' 'Tables' 'Storage' 'Furnishings' 'Art'
 'Phones' 'Binders' 'Appliances' 'Paper' 'Accessories' 'Envelopes'
 'Fasteners' 'Supplies' 'Machines' 'Copiers']
```

```
[ ] MissingValues_error_order
```

	0
Row_ID	0
Order_ID	0
Order_Date	0
Ship_Date	0
Ship_Mode	0
Status	338

dtype: int64

```
[ ] print(Order['Ship_Mode'].unique())
['Second Class' 'Standard Class' 'First Class' 'Same Day']

[ ] print(Order['Status'].unique())
['Shipped' nan 'Canceled' 'Pending']
```

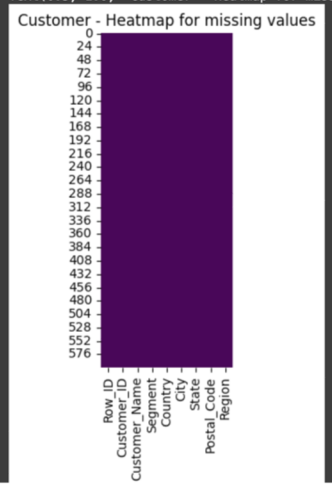
None of the attributes have entry errors except ‘Status’ which has missing values. The count of all missing values from each dataset can be specifically seen from the above images. The sum of the missing values were obtained using the ‘isnull()’ function. Entry errors may create unreliability and inconsistency in the dataset and will be difficult for the decision making.

(f) Heatmaps to check missing values

In the ‘Customer’ and ‘Product’ datasets, it can be understood that there are no missing values when analyzing the below heat map which is shown by the purple diagram. Having no missing values simplifies the data preprocessing step since no imputation or deletion of rows/columns is necessary for handling missing data.

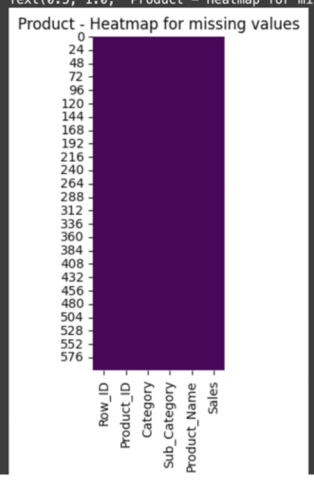
```
plt.subplot(1, 3, 1)
sns.heatmap(Customer.isnull(), cbar=False, cmap='viridis')
plt.title('Customer - Heatmap for missing values')
```

```
Text(0.5, 1.0, 'Customer - Heatmap for missing values')
```

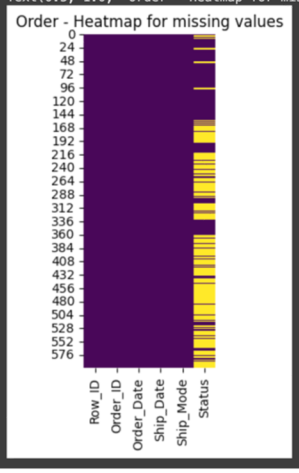



```
[ ] plt.subplot(1, 3, 2)
    sns.heatmap(Product.isnull(), cbar=False, cmap='viridis')
    plt.title('Product - Heatmap for missing values')
```

```
Text(0.5, 1.0, 'Product - Heatmap for missing values')
```



```
[ ] plt.subplot(1, 3, 3)
    sns.heatmap(Order.isnull(), cbar=False, cmap='viridis')
    plt.title('Order - Heatmap for missing values')

    Text(0.5, 1.0, 'Order - Heatmap for missing values')
```

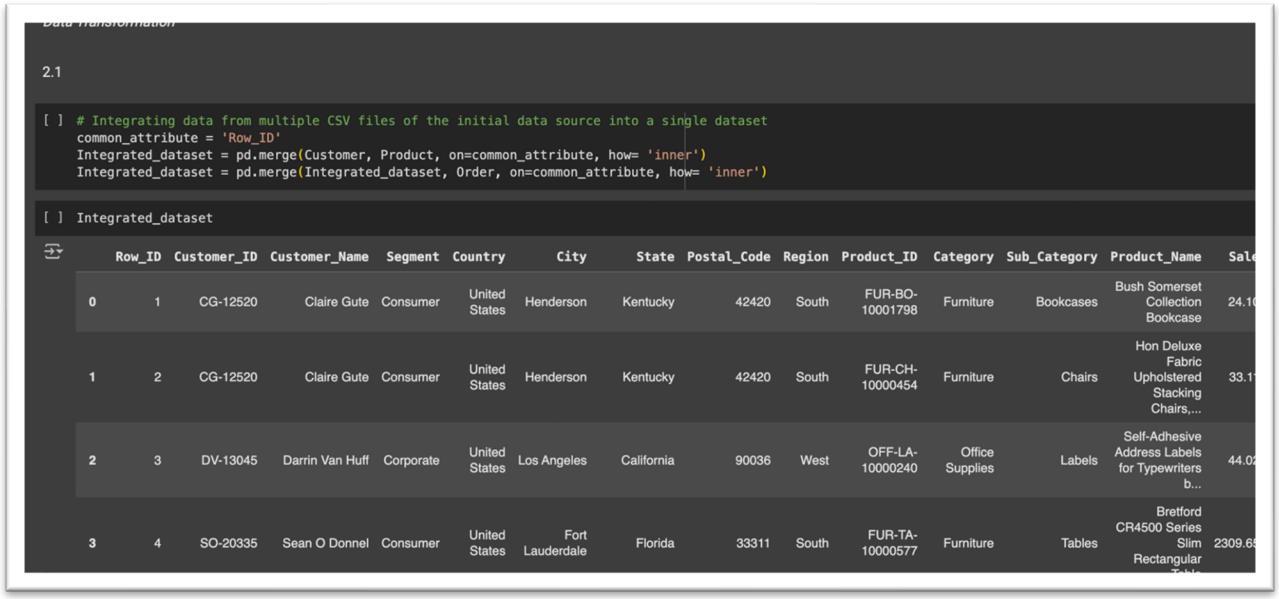


The above heatmap shows the missing values of attributes in the Order dataset. It can be seen in the heatmap that the attribute ‘Status’ has abundant missing values compared to other attributes. This shows the unreliability and inconsistency of the attribute.

Task 2: Data Transformation

Data transformation procedures such as Integrating data from multiple csv files into a single dataset, removing columns which has unwarranted records and filtering data were executed.

- Integrating data from multiple CSV files of the initial data set into a single dataset.



2.1

```
[ ] # Integrating data from multiple CSV files of the initial data source into a single dataset
    common_attribute = 'Row_ID'
    Integrated_dataset = pd.merge(Customer, Product, on=common_attribute, how= 'inner')
    Integrated_dataset = pd.merge(Integrated_dataset, Order, on=common_attribute, how= 'inner')
```

```
[ ] Integrated_dataset
```

	Row_ID	Customer_ID	Customer_Name	Segment	Country	City	State	Postal_Code	Region	Product_ID	Category	Sub_Category	Product_Name	Sale
0	1	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	24.1
1	2	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	33.1
2	3	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	44.0
3	4	SO-20335	Sean O Donnel	Consumer	United States	Fort Lauderdale	Florida	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular	2309.6

The `Customer`, `Product`, and `Order` datasets were systematically combined based on their shared `Row_ID` column. This integration resulted in a unified dataset that aligns precisely with the store's customer purchase details, where each purchase is uniquely identified by a `Purchase ID`. The final integrated dataset, named `Integrated_dataset`, contains 19 columns and 600 rows. The `Row_ID` column, unique in value and common across all three datasets, served as the primary attribute for accurately matching records.

- Removing columns of redundant features.

The ‘Customer’ and ‘Product’ datasets have no redundant values, hence there is no need to remove any columns.

2.2.a

```
[38] # Removing columns of redundant features.
     Customer_missing_values_percentage = (Customer.isnull().sum() / len(Customer)) *100
     Customer_missing_values_percentage
```

	0
Row_ID	0.0
Customer_ID	0.0
Customer_Name	0.0
Segment	0.0
Country	0.0
City	0.0
State	0.0
Postal_Code	0.0
Region	0.0

dtype: float64

```
[39] Product_missing_values_percentage = (Product.isnull().sum() / len(Customer)) *100
     Product_missing_values_percentage
```

	0
Row_ID	0.0
Product_ID	0.0
Category	0.0
Sub_Category	0.0
Product_Name	0.0
Sales	0.0

dtype: float64

```
[40] Order_missing_values_percentage = (Order.isnull().sum() / len(Customer)) *100
     Order_missing_values_percentage
```

	0
Row_ID	0.000000
Order_ID	0.000000
Order_Date	0.000000
Ship_Date	0.000000
Ship_Mode	0.000000
Status	56.333333

dtype: float64

```
[41] #Dropping colums having more than 50% missing values
     Store_dataset = Integrated_dataset.drop(['Status'], axis=1)
     Store_dataset
```

	Row_ID	Customer_ID	Customer_Name	Segment	Country	City	State	Postal_Code	Region	Product_ID	Category	Sub_Category	Product_Name	Sale
0	1	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	24.1
1	2	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	33.1
2	3	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	44.0
3	4	SO-20335	Sean O Donnel	Consumer	United States	Fort Lauderdale	Florida	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	2309.6
4	5	SO-20335	Sean O Donnel	Consumer	United States	Fort Lauderdale	Florida	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold N Roll Cart System	1090.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
595	596	LR-16915	Lena Radford	Consumer	United States	San Diego	California	92024	West	TEC-CO-10004115	Technology	Copiers	Sharp AL-1530CS Digital	1199.9

Columns that did not add significant informational value were removed after identifying those with less than 50% of the required data. This step was essential for simplifying the dataset and eliminating unnecessary complexity. For instance, the ‘Status’ column from the ‘Order’ dataset had over 50% missing values and was therefore excluded from the dataset.

- Filtering data

```
[42] # Filtering data

Filtered_Data_Home_Office = Store_dataset[Integrated_dataset['Segment'] == 'Home Office']
Filtered_Data_Home_Office
```

	Row_ID	Customer_ID	Customer_Name	Segment	Country	City	State	Postal_Code	Region	Product_ID	Category	Sub_Category	Product_Name	Sales	Order
14	15	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	Texas	76106	Central	OFF-AP-10002311	Office Supplies	Appliances	Holmes Replacement Filter for HEPA Air Cleaner...	14.030	US-20 118
15	16	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	Texas	76106	Central	OFF-BI-10000756	Office Supplies	Binders	Storex DuraTech Recycled Plastic Frosted Binders	1618.370	US-20 118
34	35	MA-17560	Matt Abelman	Home Office	United States	Houston	Texas	77095	Central	OFF-PA-10000249	Office Supplies	Paper	Easy-staple paper	14.400	CA-20 107
37	38	SN-20710	Steve Nguyen	Home Office	United States	Houston	Texas	77041	Central	OFF-EN-10002986	Office Supplies	Envelopes	#10-4 1/8 x 9 1/2 Premium Diagonal Seam Envelopes	14.427	CA-20 117
38	39	SN-20710	Steve Nguyen	Home Office	United States	Houston	Texas	77041	Central	FUR-BO-10002545	Furniture	Bookcases	Atlantic Metals Mobile 3-Shelf Bookcases, Cust...	86.620	CA-20 117
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Data was filtered using the `Segment` attribute (from the Customer dataset) to identify products purchased specifically for home office use. This transformation will help estimate the store's product demand and the revenue generated from customers purchasing items for their home office needs.

```
[43] Filtered_Data_Binders = Store_dataset[Integrated_dataset['Sub_Category'] == 'Binders']
Filtered_Data_Binders
```

	Row_ID	Customer_ID	Customer_Name	Segment	Country	City	State	Postal_Code	Region	Product_ID	Category	Sub_Category	Product_Name	Sales
8	9	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	California	90032	West	OFF-BI-10003910	Office Supplies	Binders	DXL Angle-View Binders with Locking Rings by S...	7.080
13	14	IM-15070	Irene Maddox	Consumer	United States	Seattle	Washington	98103	West	OFF-BI-10003656	Office Supplies	Binders	Fellowes PB200 Plastic Comb Binding Machine	155.250
15	16	HP-14815	Harold Pawlan	Home Office	United States	Fort Worth	Texas	76106	Central	OFF-BI-10000756	Office Supplies	Binders	Storex DuraTech Recycled Plastic Frosted Binders	1618.370
20	21	ZD-21925	Zuschuss Donatelli	Consumer	United States	San Francisco	California	94109	West	OFF-BI-10002215	Office Supplies	Binders	Wilson Jones Hanging View Binder, White, 1	124.950
25	26	EH-13945	Eric Hoffmann	Consumer	United States	Los Angeles	California	90049	West	OFF-BI-10001634	Office Supplies	Binders	Wilson Jones Active Use Binders	6.208
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
560	561	LP-17080	Liz Pelletier	Consumer	United States	San Francisco	California	94110	West	OFF-BI-10000069	Office Supplies	Binders	GBC Prepunched Paper, 19-Hole, for Binding Sys	84.056

Data was filtered using the `Sub_Category` attribute (from the Customer dataset) to identify customers who purchased 'Binders'. This transformation will assist in estimating the store's demand for binders and the revenue generated from these purchases.

- Encoding of region data

```
[44] Encoded_data = pd.get_dummies(Store_dataset, columns=['Region'], prefix=['Region'])
```

```
[45] Encoded_data
```

_Code	Product_ID	Category	...	Product_Name	Sales	Order_ID	Order_Date	Ship_Date	Ship_Mode	Region_Central	Region_East	Region_South	Region_West
42420	FUR-BO-10001798	Furniture	...	Bush Somerset Collection Bookcase	24.100	CA-2017-152156	8/11/2017	11/11/2017	Second Class	False	False	True	False
42420	FUR-CH-10000454	Furniture	...	Hon Deluxe Fabric Upholstered Stacking Chairs,...	33.110	CA-2017-152156	8/11/2017	11/11/2017	Second Class	False	False	True	False
90036	OFF-LA-10000240	Office Supplies	...	Self-Adhesive Address Labels for Typewriters b...	44.020	CA-2017-138688	12/6/2017	16/06/2017	Second Class	False	False	False	True
33311	FUR-TA-10000577	Furniture	...	Bretford CR4500 Series Slim Rectangular Table	2309.650	US-2016-108966	11/10/2016	18/10/2016	Standard Class	False	False	True	False
33311	OFF-ST-10000760	Office Supplies	...	Eldon Fold N Roll Cart System	1090.782	US-2016-108966	11/10/2016	18/10/2016	Standard Class	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...

In this data transformation process, the Region data was encoded to make it suitable for analysis and machine learning purposes. Encoding converts categorical (text-based) data into numerical format, which is often required for models to interpret and work with categorical information effectively. The Region column contains categorical values representing different geographic areas. Since numerical models and many analytical tools process numeric data more effectively than text, encoding was necessary to convert the Region values into a format that retains the original information in a machine-readable form.

- Aggregating purchase data

```
[180] # Aggregating purchase data

    Average_spending_per_customer = Store_dataset.groupby('Customer_ID')['Sales'].mean().reset_index()
    Average_spending_per_customer = Average_spending_per_customer.rename(columns={'Sales': 'Average_Spending'})
    Average_spending_per_customer
```

	Customer_ID	Average_Spending
0	AA-10375	16.768000
1	AA-10480	151.720000
2	AB-10060	79.727800
3	AC-10420	288.835000
4	AD-10180	842.468000
...	...	...
233	VP-21730	342.370000
234	VW-21775	147.318200
235	XP-21865	25.920000
236	ZC-21910	33.177250
237	ZD-21925	98.243333

238 rows × 2 columns

```
[181] Total_purchase_per_customer = Store_dataset.groupby('Customer_ID')['Sales'].sum().reset_index()
    Total_purchase_per_customer = Total_purchase_per_customer.rename(columns={'Sales': 'Total_Purchases'})
    Total_purchase_per_customer
```

	Customer_ID	Total_Purchases
0	AA-10375	16.768
1	AA-10480	151.720
2	AB-10060	398.639
3	AC-10420	577.670
4	AD-10180	4212.340
...	...	...
233	VP-21730	342.370
234	VW-21775	736.591
235	XP-21865	25.920
236	ZC-21910	132.709
237	ZD-21925	294.730

238 rows × 2 columns

To better understand customer purchasing behavior, the purchase data was aggregated using the '.mean()' and '.sum()' functions, grouping records by Customer_ID to calculate key metrics. For example, total purchase amounts and average spending per customer were calculated. This aggregation process simplifies the dataset and allows for analysis at a higher level, making it easier to identify trends, such as high-spending customers. Aggregating data in this way provides valuable insights into overall demand patterns and supports more informed decision-making for inventory and sales strategies.

- Merging aggregated purchase data to the store dataset

```
[182] Store_dataset = pd.merge(Store_dataset, Average_spending_per_customer, on='Customer_ID', how='left')

[183] Store_dataset = pd.merge(Store_dataset, Total_purchase_per_customer, on='Customer_ID', how='left')

[184] Store_dataset
```

:ry	City	State	Postal_Code	Region	Product_ID	Category	Sub_Category	Product_Name	Sales	Order_ID	Order_Date	Ship_Date	Ship_Mode	Average_S
ted ites	Henderson	Kentucky	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	24.100	CA-2017-152156	8/11/2017	11/11/2017	Second Class	
ted ites	Henderson	Kentucky	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	33.110	CA-2017-152156	8/11/2017	11/11/2017	Second Class	
ted ites	Los Angeles	California	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	44.020	CA-2017-138688	12/6/2017	16/06/2017	Second Class	
ted ites	Fort Lauderdale	Florida	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	2309.650	US-2016-108966	11/10/2016	18/10/2016	Standard Class	
ted ites	Fort Lauderdale	Florida	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold N Roll Cart System	1090.782	US-2016-108966	11/10/2016	18/10/2016	Standard Class	

Using the function ‘pd.merge’, the average spending per customer and total purchases per customer columns have been merged to the Store dataset.

- Generating metadata of the store dataset

```
[186] Store_datapoints

600

[187] Store_attributes

['Row_ID',
 'Customer_ID',
 'Customer_Name',
 'Segment',
 'Country',
 'City',
 'State',
 'Postal_Code',
 'Region',
 'Product_ID',
 'Category',
 'Sub_Category',
 'Product_Name',
 'Sales',
 'Order_ID',
 'Order_Date',
 'Ship_Date',
 'Ship_Mode',
 'Average_Spending',
 'Total_Purchases']
```

```
[188] Store_att_type
```

	0
Row_ID	int64
Customer_ID	object
Customer_Name	object
Segment	object
Country	object
City	object
State	object
Postal_Code	int64
Region	object
Product_ID	object
Category	object
Sub_Category	object
Product_Name	object
Sales	float64
Order_ID	object
Order_Date	object
Ship_Date	object
Ship_Mode	object

```
[190] #Calculating basic statistics for numerical and categorical features
      #Sales is the only numerical feature where the basic statistics can be calculated

      Numerical_statistics = Store_dataset[['Sales','Average_Spending','Total_Purchases']].describe()
      Numerical_statistics
```

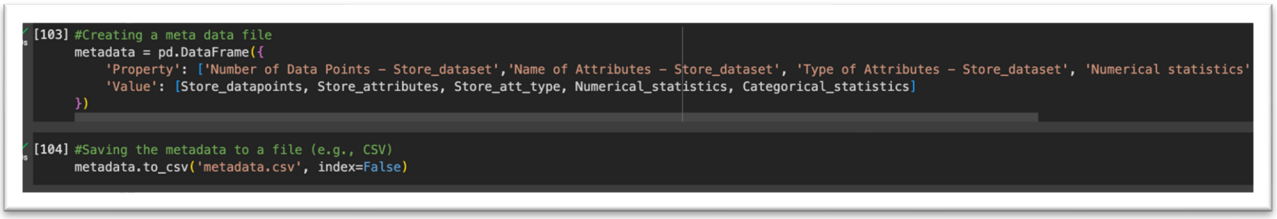
	Sales	Average_Spending	Total_Purchases
count	600.000000	600.000000	600.000000
mean	224.382289	224.382289	849.763738
std	512.768372	284.461615	1097.536469
min	1.080000	1.744000	1.744000
25%	15.552000	42.519950	132.709000
50%	46.990000	147.318200	437.763000
75%	195.783000	271.688800	1125.120000
max	6354.950000	1979.928000	6412.770000

```
[95] #Calculating basic statistics for categorical features
     Categorical_statistics = Store_dataset.describe(include=['object'])
     Categorical_statistics
```

	Customer_ID	Customer_Name	Segment	Country	City	State	Region	Product_ID	Category	Sub_Category	Product_Name	Order_ID	Order_Date	Ship_Date
count	600	600	600	600	600	600	600	600	600	600	600	600	600	600
unique	238	238	3	1	129	38	4	517	3	17	518	280	234	247
top	TB-21520	Tracy Blumstein	Consumer	United States	New York City	California	West	OFF-PA-10001970	Office Supplies	Binders	Panasonic Kx-TS550	CA-2018-117457	1/12/2018	15/06/2017
freq	11	11	344	600	51	124	203	4	355	79	3	9	13	10

The above codes generate metadata for the store dataset such as the number of datapoints, attribute types, attribute names, numerical and categorical statistics.

- Creating the metadata file of the store dataset



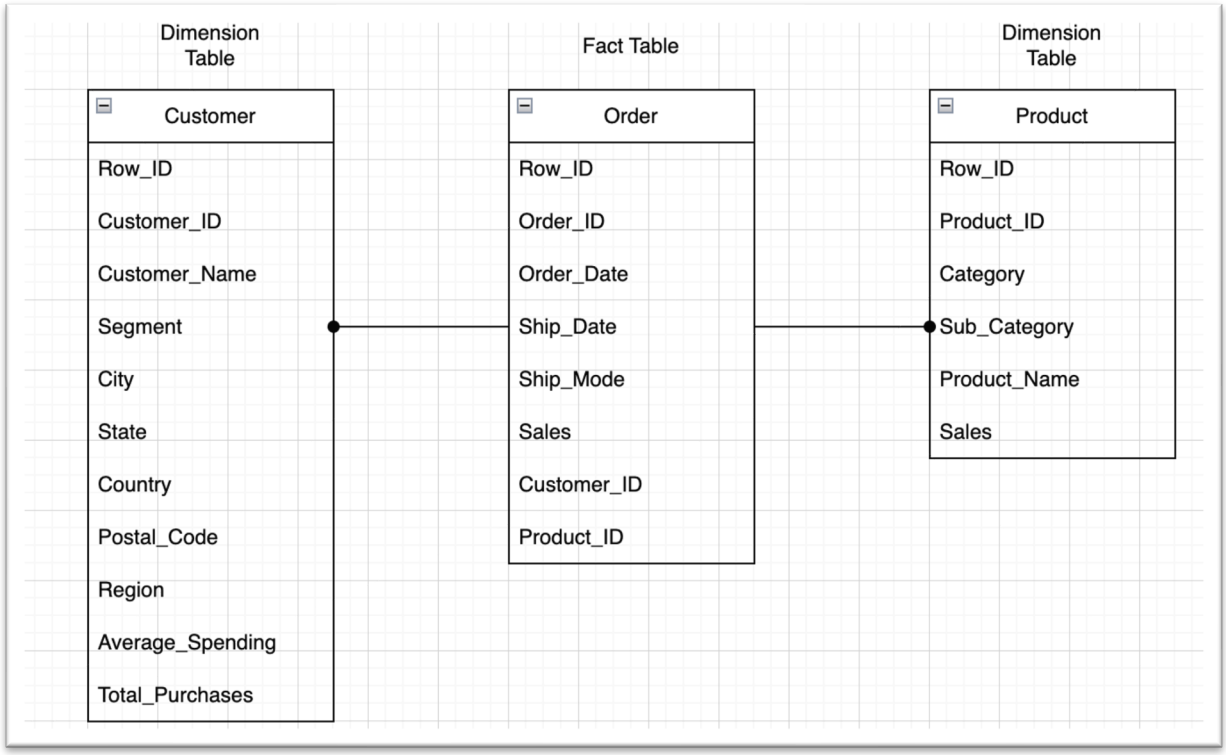
```
[103] #Creating a meta data file
      metadata = pd.DataFrame({
          'Property': ['Number of Data Points - Store_dataset','Name of Attributes - Store_dataset', 'Type of Attributes - Store_dataset', 'Numerical statistics'
          'Value': [Store_datapoints, Store_attributes, Store_att_type, Numerical_statistics, Categorical_statistics]
      })

[104] #Saving the metadata to a file (e.g., CSV)
      metadata.to_csv('metadata.csv', index=False)
```

The meta data file consists of the number of data points, names and types of the attributes, the basic numerical and categorical statistics.

Task 3: Data Loading

- Dimensional model

I implemented a star schema dimensional model to structure the data for efficient analysis. The star schema is designed with a central fact table surrounded by related dimension tables, allowing for straightforward querying and aggregation.



This star schema design simplifies data queries and improves performance for analytic tasks. Using the star schema model ensures that the data is organized in a way that supports efficient analysis and reporting on customer purchase behaviors and product demand.

- Relational database

The data loading task involved transferring the transformed and cleaned dataset into a relational database to enable efficient storage, retrieval, and analysis. After finalizing the dataset, a suitable database schema was designed, defining tables and their relationships to store the data logically and coherently. This schema setup was crucial to ensure that the database could support queries and analysis without redundancy or inconsistency.

For the loading process, a Python script was implemented to automate the data insertion. Using libraries like SQLite3, the script connected to the database and populated the tables with the cleaned data in an efficient, structured manner. This step streamlined data storage, making it easy to run queries and further analyze or visualize the data as needed.

By organizing the data in a relational database, the data loading task provided a reliable foundation for further analysis, supporting the workflow from raw data to a structured, query-ready state.

Task 3

```
[98] import sqlite3 as sq3
```

Double-click (or enter) to edit

```
[99] # connect to database
     conn = sq3.connect('Store_data.db')
     cur = conn.cursor()
```

```
[100] Store_dataset.to_sql("Store_data", conn, if_exists = 'replace', index=False)
```

```
600
```

```
[197] # print all data of the Customer table
      df1 = pd.read_sql('SELECT * FROM Store_data', conn)
      df1
```

;tal_Code	Region	Product_ID	Category	Sub_Category	Product_Name	Sales	Order_ID	Order_Date	Ship_Date	Ship_Mode	Average_Spending	Total_Purchases
42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	24.100	CA-2017-152156	8/11/2017	11/11/2017	Second Class	28.605	57.210
42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	33.110	CA-2017-152156	8/11/2017	11/11/2017	Second Class	28.605	57.210
90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	44.020	CA-2017-138688	12/6/2017	16/06/2017	Second Class	44.020	44.020
33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	2309.650	US-2016-108966	11/10/2016	18/10/2016	Standard Class	1700.216	3400.432
33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold N Roll Cart System	1090.782	US-2016-108966	11/10/2016	18/10/2016	Standard Class	1700.216	3400.432
...	...	...	...	...	...	...	...	...	...	...	...	...
92024	West	TEC-CO-	Technology	Copiers	Sharp AL-1530CS Digital	1199.976	CA-2015-	8/8/2015	15/08/2015	Standard	512.454	2049.816

```
[102] conn.commit()
      conn.close()
```

Self-Reflection

This coursework on data engineering introduced me to the process of building a data pipeline, from scratch to data storage. Initially, I experienced some challenges in data cleaning, especially with missing values and removing non-essential columns. Every case of data cleansing posed a dilemma regarding the merits and demerits of such practices regarding the quality of analysis. In the process, I understood how one needs to deal with data - completeness as much as accuracy.

My most extensive experience came during the data transformation stage when I had to merge the datasets and perform some feature engineering. For instance, I had to aggregate the purchasing information and encode some categorical attributes such as the `Region` one, which taught me a lot about how to analyze data deeply. These changes required not only technical know-how but also wits as to the usefulness of the dataset to the business, such as customer demand forecasting and pattern analysis.

Loading data into the system was one of the most challenging yet intriguing activities; I created a database schema and populated it with Tables containing the cleansed and transformed data. Also, data storage is essential to any winning pipeline, especially when the respective datasets are large. Creating data load scripts for populating a relational database made me understand why data engineers adhere to structure.

In general, this coursework developed my competencies concerning data cleaning, transforming, and loading, and the significance of documentation was highlighted. It made me appreciate that the whole pipeline has the beginning and the end, significantly affecting the data's quality and usability. Going through this has improved my skills and made me appreciate the scope of where data engineering sits in practice.